

# Squeezing bottlenecks: exploring the limits of autoencoder semantic representation capabilities

Parth Gupta<sup>1</sup>, Rafael E. Banchs<sup>2</sup> and Paolo Rosso<sup>1</sup>

<sup>1</sup>Natural Language Engineering Lab- PRHLT  
Department of Information Systems and Computation  
Universitat Politècnica de València, Spain  
{pgupta,prossso}@dsic.upv.es

<sup>2</sup>Institute for Infocomm Research, Singapore  
rembanchs@i2r.a-star.edu.sg

## Abstract

We present a comprehensive study on the use of autoencoders for modelling text data, in which (differently from previous studies) we focus our attention on the following issues: *i*) we explore the suitability of two different models bDA and rsDA for constructing deep autoencoders for text data at the sentence level; *ii*) we propose and evaluate two novel metrics for better assessing the text-reconstruction capabilities of autoencoders; and *iii*) we propose an automatic method to find the critical bottleneck dimensionality for text language representations (below which structural information is lost).

## 1 Introduction

One of the major hurdles in comparing text in vector space models (VSM) is to deal with problems like *synonymy* and *polysymy*. Usually in vector space, the documents are composed of thousands of dimensions. In addition to high computational complexity, many meaningful associations between terms are shadowed by large dimensions. There are models which try to solve this problem *e.g.* pseudo relevance feedback (PRF) and explicit semantic analysis (ESA) (Xu and Croft, 1996; Gabrilovich and Markovitch, 2007). Other category of attempts to solve this problem comprise of dimensionality reduction techniques.

The goal of dimensionality reduction techniques is to transform high dimensional data ( $\mathbb{R}^n$ ) into a much lower dimension representation ( $\mathbb{R}^m$ ) pertaining the inherent structure of the original data where  $m \ll n$ . One such widely used approach is latent semantic indexing (LSI) which extracts a low rank approximation of a term-document matrix by means of principal component analysis (PCA) (Deerwester et al., 1990). There are some advanced approaches like prob-

abilistic latent semantic analysis (PLSA) and latent dirichlet allocation (LDA) which observe the distribution of latent topics for the given documents (Hofmann, 1999; Blei et al., 2003).

Dimensionality reduction techniques are also prominent while estimating the similarity between text across languages. Associations of terms and documents across languages in such techniques are learnt by means of parallel or comparable text (Nie et al., 1999; Banchs and Kaltenbrunner, 2008; Platt et al., 2010).

Dimensionality reduction techniques can broadly be categorised in two classes: linear and non-linear. Usually, non-linear techniques can find more compact representations of the data compared to their linear counterparts (Hinton and Salakhutdinov, 2006). If there exists statistical dependence among the principal components of PCA, or principal components have non-linear dependencies, PCA would require a larger dimensionality to properly represent the data when compared to non-linear techniques.

On the other hand, although non-linear projection methods such as multidimensional scaling (MDS) give a way to obtain much better representations for mono and cross-language similarity estimation, it is a transductive method (Cox and Cox, 2001; Banchs and Kaltenbrunner, 2008). It means MDS does not provide an operator to project the unseen data into the target low dimensional space like the resulting projection matrix in the case of PCA.

Lately, dimensionality reduction techniques based on deep-learning have become very popular, especially deep autoencoders (DA). Deep autoencoders can extract highly useful and compact features from the structural information of the data. Deep autoencoders have proven to be very effective in learning reduced space representations of the data for similarity estimation, *i.e.* similar documents tend to have similar abstract representations (Hinton and Salakhutdinov, 2006; Salakhut-

dinov and Hinton, 2009a). Deep learning is inspired by biological studies which state the brain has a deep architecture. Despite their high suitability to the task, deep-learning did not find much audience because of convergence issues until Hinton and Salakhutdinov (2006) gave a way to initialise the network parameters in a good region for finding optimal solutions.

Deep learning based dimensionality reduction techniques are quite popular for NLP tasks like sentiment prediction (Socher et al., 2011), part-of-speech tagging, chunking, named entity recognition, semantic role labeling (Collobert et al., 2011) and semantic compositionality (Socher et al., 2012; Socher et al., 2013). Unlike such tasks where sequential information among the words is important, information retrieval like similarity estimation based models such as ours use bag-of-words representation of the text (Yih et al., 2011; Platt et al., 2010; Salakhutdinov and Hinton, 2009a).

Although deep learning techniques are in vogue, there still exist some important open questions. In most of the studies involving the use of these techniques for dimensionality reduction, the qualitative analysis of projections is never presented. This makes the assessment of the reliability of learning very difficult. Typically, the reliability of the autoencoder is estimated based on its reconstruction capability.

The objective of this work is to propose a novel framework for evaluating the quality of the dimensionality reduction task based on the merits of the application under consideration: the representation of text data in low dimensional spaces.

Concretely, our proposed framework is comprised of two metrics, *structure preservation index (SPI)* and *similarity accumulation index (SAI)*, which capture different aspects of the autoencoder’s reconstruction capability like the structural distortion of the data and similarities among the reconstructed vectors. In this way, our proposed framework gives better insight of the autoencoder performance allowing for conducting better error analysis and evaluation, and, as explained below, these metrics also provides a better means for estimating the adequate size of critical bottleneck dimensions.

We carry out the experiments of dimensionality reduction of text at sentence level and assess the suitability of two types of deep autoencoders. We report some interesting findings at the architectural level of the specific problem of modelling text at the sentence level.

The rest of the paper is structured as follows. A brief introduction to deep autoencoders is given in Section 2. Section 3 gives details about the analysis framework of the autoencoder learning, ex-

periments and results. The discussion on critical bottleneck dimensionality and an automatic way to estimate it is given in Section 4. Finally, we present the conclusions and future directions of this work in Section 5.

## 2 Models

In this section we describe the models we have considered for performing dimensionality reduction of text data. First, we provide a brief introduction to autoencoders. Then, in sub-section 2.1, we present the binary deep autoencoder model (bDA); and, in sub-section 2.2, we describe the replicated softmax deep autoencoder (rsDA). Finally, in sub-section 2.3, we discuss the training procedure in detail.

Both of the considered models differ in the way they model the text data. While the binary deep autoencoder models the presence of the term into the document (*binary*), the replicated softmax deep autoencoder directly models the count of the term (i.e., *term frequency*) in the document.

The autoencoder is indeed a network which tries to learn an approximation of the identity function so as the output is similar to input. The input and output dimensions of the network are the same ( $n$ ). The autoencoder approximates the identity function in two steps: *i*) reduction, and *ii*) reconstruction. The reduction step takes the input  $x \in \mathbb{R}^n$  and maps it to  $y \in \mathbb{R}^m$  where  $m < n$  which can be seen as a function  $y = g(x)$  with  $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$ . On the other hand, the reconstruction step takes the output of the reduction step  $y$  and maps it to  $\hat{x} \in \mathbb{R}^n$  in such a way  $\hat{x} \approx x$  which is considered as a  $\hat{x} = f(y)$  with function  $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ . The full autoencoder can be seen as  $f(g(x)) \approx x$ .

In a neural network based implementation of the autoencoder, the visible layer corresponds to the input  $x$  and the hidden layer corresponds to  $y$ . There are two variants of autoencoders: *i*) with a single hidden layer, and *ii*) with multiple hidden layers. If there is only one single hidden layer, the optimal solution remains the PCA projection even with the added non-linearities in the hidden layer (Bourlard and Kamp, 1988). The PCA limitations are overcome by stacking multiple encoders, constituting what is called a deep architecture. This deep construction is what leads to a truly non-linear and powerful reduced space representation (Hinton and Salakhutdinov, 2006). The deep architecture is constituted by stacking multiple restricted boltzmann machines (RBM) on top of each other as shown in Fig. 1.

Each RBM is a two-layer bipartite network with a visible layer (**v**) and a hidden layer (**h**). Both layers are connected through symmetric weights (**w**). Usually the hidden units correspond to *latent*

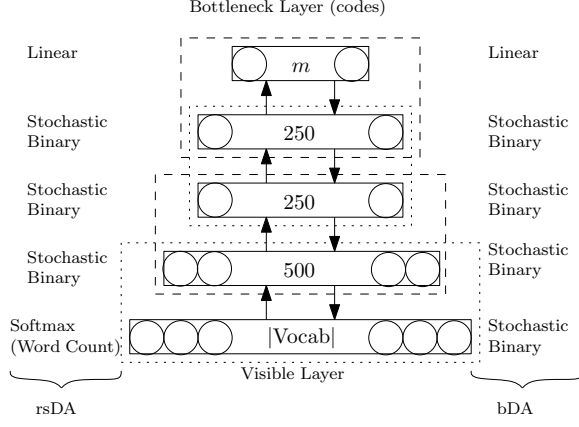


Figure 1: Architecture of the deep autoencoders. The binary and replicated softmax deep autoencoders are denoted as bDA and rsDA.  $|\text{Vocab}|$  is the size of vocabulary at the input layer.

variables. It is very easy to sample the data from visible to hidden layer and vice-versa. The two models we consider here, bDA and rsDA, primarily differ in the bottom-most RBM, i.e. the way they model the input data. In a nutshell, in the case of bDA, the bottom-most RBM is a standard RBM with stochastic binary (visible and hidden) layers; while, in the case of rsDA, the bottom-most RBM is based on the replicated softmax model (RSM) (Salakhutdinov and Hinton, 2009b).

## 2.1 Stochastic Binary RBM

Stochastic binary RBMs have both, visible and hidden, layers as stochastic binary with sigmoid non-linearity. Let visible units  $\mathbf{v} \in \{0, 1\}^n$  be binary bag-of-words representation of text documents and hidden units  $\mathbf{h} \in \{0, 1\}^m$  be the hidden latent variables. The energy of the state  $\{\mathbf{v}, \mathbf{h}\}$  is as follows,

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{i=1}^n a_i v_i - \sum_{j=1}^m b_j h_j - \sum_{i,j} v_i h_j w_{ij} \quad (1)$$

where  $v_i, h_j$  are the binary states of visible unit  $i$  and hidden unit  $j$ ,  $a_i, b_j$  are their biases and  $w_{ij}$  is the weight between them.

Then, it becomes easy to sample the data in both directions as shown below,

$$p(v_i = 1 | \mathbf{h}) = \sigma(a_i + \sum_j h_j w_{ij}) \quad (2)$$

$$p(h_j = 1 | \mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}) \quad (3)$$

where  $\sigma(x) = 1/(1 + \exp(-x))$  is the logistic sigmoid function.

## 2.2 Replicated Softmax RBM

The Replicated Softmax RBM is based on the Replicated Softmax Model (RSM) proposed by Salakhutdinov and Hinton (2009b).

Let  $\mathbf{v} \in \{1, \dots, K\}^D$ , where  $K$  is the vocabulary size,  $D$  is size of the document and let  $\mathbf{h} \in \{0, 1\}^m$  be stochastic binary hidden latent variables. Considering a document with length  $D$ , the energy of the state  $\{\mathbf{v}, \mathbf{h}\}$  is defined as,

$$E(\mathbf{v}, \mathbf{h}) = - \sum_{k=1}^K \hat{v}^k a^k - D \sum_{j=1}^m b_j h_j - \sum_{k,j} W_j^k h_j \hat{v}^k \quad (4)$$

where,  $\hat{v} = \sum_{i=1}^D v_i^k$  denotes the count data for the  $k^{th}$  term.

In RSM, the visible layer is softmax with multinomial visible units which represents the probability distribution of the word-count. It is sampled  $D$  times by using multinomial sampling to recover the original word-count data. Another distinction of this model is scaling of the bias terms of the hidden layer which gives a way to handle the documents of different lengths. In this case, the visible and hidden units are updated as shown below,

$$p(v_i^k = 1 | \mathbf{h}) = \frac{\exp(b_i^k + \sum_j h_j W_{ij}^k)}{\sum_{q=1}^K \exp(b_i^q + \sum_j h_j W_{ij}^q)} \quad (5)$$

$$p(h_j = 1 | \mathbf{V}) = \sigma(a_j + \sum_{i=1}^D \sum_{k=1}^K v_i^k W_{ij}^k) \quad (6)$$

## 2.3 Training of Autoencoders

Autoencoders are typically trained in two steps: *i*) greedy layerwise pre-training, and *ii*) fine-tuning of the parameters to learn the identity approximation of the input data.

### 2.3.1 Pre-training

In this step, each RBM is trained greedily using contrastive divergence (CD) learning (Hinton, 2002). Here the RBMs are trained one by one starting from the bottom-most RBM. The bottom-most RBM directly takes the input data while the upper RBMs take the output  $p(\mathbf{h} | \mathbf{v})$  of the RBM below which is already trained. We use the structure of the autoencoder 500-250-250-m as shown in Fig. 1. We train each RBM using  $CD_1$  learning for 50 epoch where  $CD_1$  refers to CD with 1 step of alternating Gibbs sampling (Hinton, 2002).

### 2.3.2 Fine-tuning

Once the RBMs are trained layer-wise, the autoencoder is unrolled as shown in Fig. 2. The stochastic binary activities of the feature layers is replaced by the real-valued probabilities and the

input data is backpropagated through the network to fine-tune the parameters of the entire network. We calculate the cross-entropy error ( $e$ ) between  $\hat{x} = f(g(x))$  and  $x$  as shown below and backpropagate it through the entire network.

$$e = - \sum_{i=1}^n [x_i \log(\hat{x}_i) + (1 - x_i) \log(1 - \hat{x}_i)] \quad (7)$$

In case of bDA the binary input data is used to calculate  $e$ . While for rsDA, the word-count input vectors are divided by the document length ( $D$ ) to represent probability distribution which together with softmax output layer is used to calculate  $e$ .

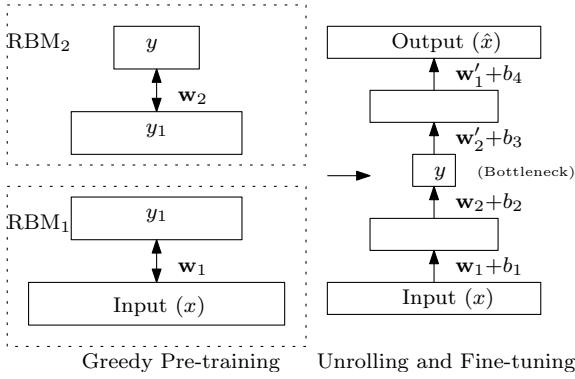


Figure 2: **Left panel:** pre-training the stacked RBMs where upper RBMs take output of the lower RBM. **Right panel:** After pre-training the structure is “unrolled” to create a multi-layer autoencoder which is fine-tuned by backpropagation to perform  $\hat{x} \approx x$ .

### 3 Qualitative Analysis and Metrics

In this section we describe the proposed metrics used for comparing the bDA and rsDA models. Subsequently, we present the comparative analysis of the two models.

The quality of the projections and the sufficiency of dimension  $m$  are measured by the autoencoder’s reconstruction ability. Unfortunately, the mean squared error between the input  $x$  and its reconstruction  $\hat{x}$ , referred as *reconstruction error*, is a poor measure to estimate it. It neither gives any details about the quality of the reconstructions in terms of text data representation nor the degree to which the structure of the data (distance between documents in original space) is preserved in the reconstruction space. Moreover, it is difficult to justify the adequacy of bottleneck dimension  $m$  by simply using the *reconstruction error*.

In literature, when autoencoders are used for dimensionality reduction for text data, most of the

time, the quality of the algorithm is measured in terms of the accuracy of the end-task which may be text categorisation (Hinton and Salakhutdinov, 2006), information retrieval (Salakhutdinov and Hinton, 2009a), or topic modeling (Salakhutdinov and Hinton, 2009b). A shortcoming of this approach is that there is no way to estimate the full potential, or the upper bound, of the algorithm performance. On the other hand, in the case of poor results, it becomes tougher to decide whether the training was proper or not.

As already mentioned before, in this work we propose two new metrics: *i) structure preservation index (SPI)*, and *ii) similarity accumulation index (SAI)*, which are intended to capture different aspects of the autoencoder’s reconstruction capability, like the structural distortion and semantic similarity of the reconstructed vectors with respect to the original ones. Considering these two metrics, along with the *reconstruction error*, allows for a much better assessment of confidence regarding the quality of the network training process and its performance.

**Structure Preservation Index (SPI):** Consider the input data as  $X$  where each row  $x_i$  corresponds to the vector space representation of the  $i^{th}$  document and  $\hat{X}$  is its corresponding reconstruction.  $X$  and  $\hat{X}$  are  $pxn$  matrices where  $p$  is the total number of documents in the input data and  $n$  is the vocabulary size. Compute matrix  $D$  for  $X$  such that  $D_{ij}$  is the cosine similarity score between  $i^{th}$  and  $j^{th}$  rows of  $X$ . Similarly calculate  $\hat{D}$  for  $\hat{X}$ .  $D$  and  $\hat{D}$  can be seen as similarity matrices of the original data and its reconstruction, respectively, where  $D_{ij} = \hat{D}_{ij} = 1, \forall i = j$ . The SPI is calculated as follows,

$$SPI = \frac{1}{p^2} \sum_{ij} ||D_{ij} - \hat{D}_{ij}||^2 \quad (8)$$

Notice that according to this definition, SPI captures the structural distortion incurred by the  $f(g(X))$  process. Ideally, SPI should be zero.

**Similarity Accumulation Index (SAI):** Different from SPI, which assesses structural distortion, SAI attempts to capture the quality of the reconstructed vectors by measuring the cosine similarity between each original vector and its reconstructed version. Indeed, this verifies the preservation of the relative strength of the vector-dimensions in the reconstruction.

SAI is computed by the normalised accumulation of cosine similarities between each input document and its reconstruction. Ideally, SAI should

be one.

$$\text{SAI} = \frac{1}{p} \sum_{i=1}^p \text{cosine}(x_i, \hat{x}_i) \quad (9)$$

### 3.1 Comparative Evaluation of Models

We carried out an experiment of dimensionality reduction for text sentences, where data sparseness plays a more critical role than in the case of full documents (dimensionality reduction applied to full documents is the case that has been mostly explored in the literature).

In this study we aim at applying autoencoder techniques at the level of sentences to open its way for sentence-centered applications, such as machine translation, text summarization and automatic dialogue response.

For this experiment, we used the Bible dataset, which contains 25122 training and 995 test sentences. All sentences were processed by a term-pipeline of stopword-removal and stemming which is referred as **Vocab**<sub>1</sub>. After that we kept only those terms which were non-numeric, at least 3-characters long and appeared in at least 5 training sentences. We refer to this filtered vocabulary as **Vocab**<sub>2</sub>. For English partition of the dataset, **Vocab**<sub>1</sub> and **Vocab**<sub>2</sub> are 8279 and 3100 respectively.

Next, we present the results for English using both models, bDA and rsDA, and present the qualitative analysis of the reconstructions with the help of the aforementioned metrics. We train both autoencoders with the structure 500-250-250-40 as described in Section 2.3. The results are presented in Table 1.

Model	RC	SPI	SAI
rsDA ( <i>pt</i> )	0.1192	0.7258	0.2132
rsDA ( <i>bp</i> )	0.0834	0.0049	0.5768
bDA ( <i>pt</i> )	8.0012	0.0712	0.3528
bDA ( <i>bp</i> )	5.4829	0.0035	0.6667

Table 1: The performance of bDA and rsDA in terms of different metrics. **RC** denotes *reconstruction error* while *pt* and *bp* denote if the model is only pre-trained and fine-tuned after pre-training, respectively.

### 3.2 Analysis and Discussion

When operating in vector space, it is important to understand the amount of distortion incurred by the network on the structure of the data during the process of  $f(g(x))$ . The network uses the *reconstruction error* during the training to update parameters but it does not give much insight about

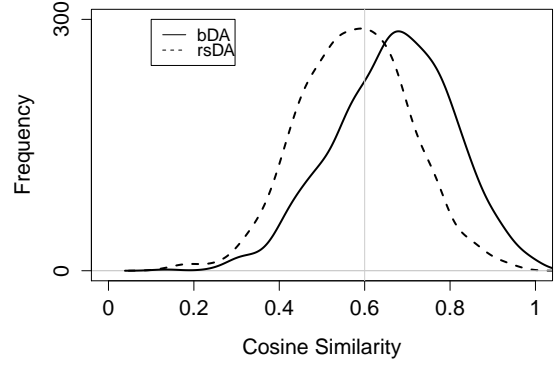


Figure 3: Histogram of cosine similarity between test samples and their reconstructions for bDA and rsDA.

the quality of the network. One more limitation of the *reconstruction error* is that it is not bounded and not comparable across different models *e.g.* bDA and rsDA. The *reconstruction error* is calculated between the softmax output and the probability distribution of terms in case of rsDA hence it is not comparable to that of bDA (see Table 1).

The two proposed metrics, SPI and SAI are both bounded by [0,1] and comparable across the models. SPI gives the measure of how the similarity structure of sentences among each other is preserved in the reconstruction space which in turn gives a measure of trustworthiness of the network for similarity estimation. Although both models show descent performance in terms of SPI after backpropagation, bDA is 28.57% better than rsDA in terms of SPI.

It is also important to assess the similarity between each input vector and its corresponding reconstruction which is captured by SAI. In terms of SAI, bDA is 15.59% better than rsDA. This is better understood in Fig. 3, where it can be noticed that, in the case of rsDA, for more than half of the test samples cosine similarity with their reconstruction is  $\leq 0.6$ . Although rsDA has been reported in the literature to better perform at the document level, our results demonstrate that bDA is a more suitable model to be used when using autoencoder representations at the sentence level. This can be explained by the fact that rsDA uses multinomial sampling to model the word-count, which happens not to be suitable at the sentence level for two reasons: *i*) most of the terms appear only once in the sentences, and *ii*) sampling the distribution of terms  $D$  times is less reliable when  $D$  is quite small which is the case in sentences compared to full documents.

Finally, as argued by Erhan et al. (2010), pre-

training helps to initialise the network parameters in a region to find optimal solution. It can clearly be noticed that pre-training is necessary but itself is not enough to put aside backpropagation.

## 4 Critical Bottleneck Dimensionality

In this section we present the analysis on the adequacy of the size of bottleneck layer. The top-most hidden layer of an autoencoder is commonly referred to as the bottleneck layer. The reconstruction ability of the autoencoder is highly related to the size of the bottleneck layer, in the sense that the smaller the size of the bottleneck layer is, the higher the loss of information is.

The reduction step of autoencoders is also called *hashing*, and because similar sentences in the projected space are near to each other, this technique is also referred to as *semantic hashing*. It is important to choose a proper size of the bottleneck layer because of two reasons: *i*) a too large size may lead to redundant dimensions and high computational cost, and *ii*) a too small size might lead to high information loss.

The adequacy of the bottleneck dimension, which we refer to as critical bottleneck dimensionality here, is rarely addressed in the literature. In this section of the study, we present an analysis on the effects of choosing different sizes for the bottleneck layer, as well as we provide an empirical method to choose the critical bottleneck dimensionality properly.

### 4.1 Metric Selection

We squeeze the bottleneck layer of the autoencoder to identify whether there was a dimensionality region at which the *reconstruction error*, SPI and SAI metrics exhibit a clear change in its behaviour. Typically, this region is referred to as the “elbow region”. We trained the autoencoder varying down the size of the bottleneck layer from 100 to 10 with step-sizes of 10. Fig. 4 shows the values of *reconstruction error*, SPI and SAI for different sizes of bottleneck layer.

As it becomes evident from the figure 4, SPI is the metric exhibiting the clearest “elbow region” pattern, hence we will use this metric for determining the critical bottleneck dimensionality. Indeed, it can be noticed that both the *reconstruction error* and SAI show a quasi-linear behaviour with almost constant slope, while SPI clearly captures that below  $m = 40$ , the network starts losing the structural information within the data. This result shows that care must be taken to select a proper bottleneck dimension and it is important not to choose the bottleneck dimension below the point where SPI changes its behaviour.

### 4.2 Identification of Critical Dimensionality

To identify the critical bottleneck dimensionality, we calculated the percentage difference between the slopes connecting consecutive bottleneck sizes in the SPI curve. This captures the point in the “elbow region” at which the slope of the SPI curve is steepest. Consider three points in SPI plot:  $a_1$ ,  $a_2$  and  $a_3$ . Let  $s_1^2$  and  $s_2^3$  be the slopes of lines connecting  $a_1 - a_2$  and  $a_2 - a_3$ , respectively. Then the percentage difference between  $s_1^2$  and  $s_2^3$  gives the steepness of the curve at point  $a_2$ . We calculate this figure for every point in the range and estimate the *critical dimensionality* at which the percentage difference is the largest. This method enables us to automatically find the adequate bottleneck dimension. The algorithmic implementation of this method is described in Fig. 5.

#### Method: Estimation of *critical* dimension

**Input:**  $A, B$

**Output:**  $C$

$A$  = set of bottleneck dimensions

$B$  = set of SPI values, where  $b_i = \text{SPI}(a_i) \in A$

$C$  = set of steepness values at each point

**for each**  $a_{i-1}, a_i, a_{i+1} \in A$

**get**  $b_{i-1}, b_i, b_{i+1} \in B$

**calc.**  $s_{i-1}^i, s_i^{i+1}$  where,

$s_{i-1}^i = \text{slope}((a_{i-1}, b_{i-1}), (a_i, b_i))$

**calc.**  $c_i = \% \text{ diff } (s_{i-1}^i, s_i^{i+1})$

**add**  $c_i$  to  $C$

**end**

**plot**  $C$

*critical dim.* = right-most large peak

Figure 5: Method to identify the *critical dimensionality* for the bottleneck layer.

## 5 Conclusions and Future Research Directions

In this work we have presented a comprehensive study on the use of autoencoders for modelling text data, in which differently from previous studies we focused our attention in the following issues:

- We explored the suitability of two different models bDA and rsDA for constructing deep autoencoder representations of text data at the sentence level.
- We proposed and evaluated two novel metrics which assess the reconstruction quality of an autoencoder with regards to the particular problem of text data representation.
- We proposed an automatic method to find the critical bottleneck dimensionality for text

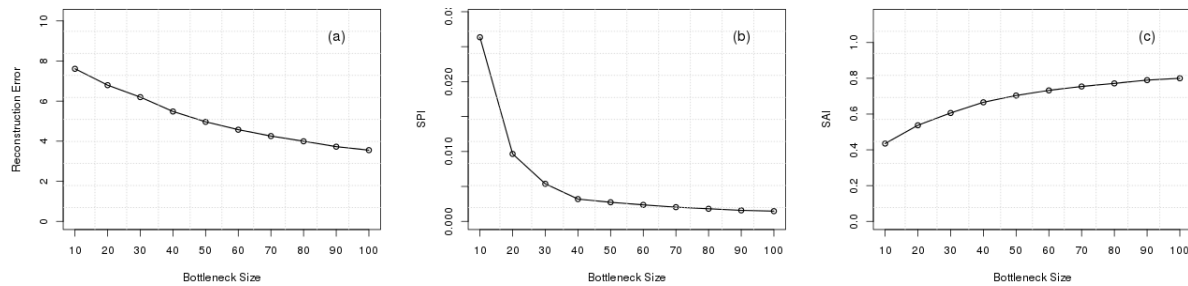


Figure 4: *Reconstruction error (a), SPI (b) and SAI (c) metrics while squeezing the bottleneck layer from 100 to 10 of bDA.*

language representation, below which structural information is lost.

As a result of this study we have found that the bDA model is most suitable for constructing and training autoencoders for handling text data at the sentence level. We also found that our defined SPI (Structure Preservation Index) metric allows for a better discrimination and identification of the critical bottleneck dimensionality.

As future work, we want to study the suitability of our proposed metrics, especially SPI, as error metric during the autoencoder fine tuning stage. If this metric can be used along with back-propagation, we envisage a new generation of text-oriented autoencoders able to provide a much better characterization of the linguistic phenomenon in text data.

## Acknowledgment

The work of the first and third authors was carried out in the framework of the WIQ-EI IRSES project (Grant No. 269180) within the FP 7 Marie Curie, the DIANA APPLICATIONS Finding Hidden Knowledge in Texts: Applications (TIN2012-38603-C02-01) project and the VLC/CAMPUS Microcluster on Multimodal Interaction in Intelligent Systems.

## References

- Rafael E. Banchs and Andreas Kaltenbrunner. 2008. Exploiting mds projections for cross-language ir. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *SIGIR*, pages 863–864. ACM.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.
- H. Bourlard and Y. Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4):291–294, September.
- R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.
- Trevor F. Cox and Michael A. A. Cox. 2001. *Multidimensional Scaling*. CRC/Chapman and Hall.
- Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407.
- Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March.
- Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artificial intelligence, IJCAI’07*, pages 1606–1611, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Geoffrey Hinton and Ruslan Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507.
- Geoffrey E. Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800.
- Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’99*, pages 50–57, New York, NY, USA. ACM.
- Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. 1999. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’99*, pages 74–81, New York, NY, USA. ACM.
- John C. Platt, Kristina Toutanova, and Wen tau Yih. 2010. Translingual document representations from

discriminative projections. In *EMNLP*, pages 251–261. ACL.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009a. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, July.

Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009b. Replicated softmax: an undirected topic model. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *NIPS*, pages 1607–1614. Curran Associates, Inc.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP-CoNLL*, pages 1201–1211.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Chris Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '96, pages 4–11, New York, NY, USA. ACM.

Wen-tau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, CoNLL '11, pages 247–256, Stroudsburg, PA, USA. Association for Computational Linguistics.